A Foundational Model of Time for Heterogeneous Clinical Databases

Amar K. Das and Mark A. Musen
Section on Medical Informatics, Stanford University School of Medicine
Stanford, California 94305-5479
das@smi.stanford.edu

Differences among the database representations of clinical data are a major barrier to the integration of databases and to the sharing of decision-support applications across databases. Prior research on resolving data heterogeneity has not addressed specifically the types of mismatches found in various timestamping approaches for clinical data. Such temporal mismatches, which include time-unit differences among timestamps, must be overcome before many applications can use these data to reason about diagnosis, therapy, or prognosis. In this paper, we present an analysis of the types of temporal mismatches that exist in databases. To formalize these various approaches to timestamping, we provide a foundational model of time. This model gives us the semantics necessary to encode the temporal dimensions of clinical data in legacy databases and to transform such heterogeneous data into a uniform temporal representation suitable for decision support. We have implemented this foundational model as an extension to our Chronus system, which provides clinical decision-support applications the ability to match temporal patterns in clinical databases. We discuss the uniqueness of our approach in comparison with other research on representing and querying clinical data with varying timestamp representations.

TEMPORAL MISMATCH IN DATABASES

Because clinical information is complex and multifaceted, developers of clinical computer applications are able to generate a wide variety of data representations to suit their needs. Such diversity in representations is readily apparent if one examines the schemas of clinical databases. Data representations can differ between databases because of the variety of conceptual data models, the naming of elements in the data schema, and the multitude of coding methods. Such variations in data representations may be necessary to satisfy the data-management needs of local database users. These differences make difficult, however, the integration and comparison of clinical data with multiple data schemas.

Overcoming the heterogeneity, or mismatches, present in database representations has become the major effort of those researchers whose goals are either (1) to integrate databases distributed across computer networks [1] or (2) to reuse decision-support applications across different databases [2]. One important type of data heterogeneity that has not been handled formally by these two types of approaches is the *temporal mismatch* created by various timestamping approaches for clinical data. Previous methods can overcome some simple types of temporal heterogeneity (such as naming differences among timestamp attributes); however, these methods do not resolve more complex mismatches that are present with timestamps at varying time units or with the encoding of temporal information as duration values (such as an age attribute). Prior methods, in fact, have yet to address the *foundational model of time* that is necessary to create a unified temporal representation for data that have temporal mismatches. Such a model of time is essential, for example, to compare an age attribute in one database schema with the birthdate timestamp in another database. Unless such temporal mismatches are resolved, users may face difficulty in integrating temporal data from multiple databases and creating computer applications that need a uniform temporal view of clinical data [3] .

In our development of the Chronus system [4] — a temporal querying method for clinical decision-support applications — we have created a temporal data model that can represent uniformly the temporal dimensions of various types of clinical data. Our temporal scheme requires that all data have interval timestamps. For instant-based data, which we call *events*, the value of the timepoints are equal, whereas for interval-based data, which we call *states*, the timepoints are the endpoints of the interval. For time-invariant data, we can assign the start and end times to constant timepoint values. Although all timepoint values in our interval-based representation must have the same time-unit level (such as minute), we have shown previously that this representation can capture the uncertainty associated with timestamps of varying time units [4] . Our proposed timestamping scheme thus provides a unified view of timestamps upon which decision-support systems can base their temporal representations [5] . More importantly, our scheme permits a *closed* set of temporal operators that applications can use to formulate temporal queries of arbitrary complexity [4].

Although Chronus can maintain a uniform view of time in temporal querying, it does not provide a method to map data in various timestamping schemes into its canonical temporal representation. In this paper, we present our method to overcome temporal mismatches. We discuss first the types of temporal mismatches that are present in clinical databases. We then establish a foundational model of time, so that we can express formally the semantic differences among various timestamping approaches. Using mapping functions that are based on this model, we have created a database

1091—8280/97/$5.00

method to transform data with heterogeneous temporal representations into a canonical timestamped format. We have implemented this method as an extension to Chronus — which we call Synchronus. In conclusion, we discuss the novelty of this approach in comparison with related approaches to modeling data with complex temporal representations.

TYPES OF TEMPORAL HETEROGENEITY

We have undertaken a review of the temporal representations that are used for the data sources of various decision-support systems or that are described in the research literature. Our analysis reveals two types of temporal heterogeneity: (1) *representational mismatch* and (2) *semantic mismatch*. We present in this section the various types of mismatches.

Representational Mismatch

Representational mismatch implies that the timestamping approaches of two sets of data have equivalent meaning, but different representations.

Temporal metrics. Many different metrics are available to measure the value of discrete timepoints. For example, we can store timestamps in the American or the European format, or we can represent time values with a 12- or 24-hour clock.

Timestamp name. The naming of timestamps may differ among data representations. For example, `PROBLEM_START` and `MEDICATION_START` are attributes with equivalent semantics, since both store the start of an interval-based datum. In a uniform temporal representation, both attributes would have the same name (such as `START_TIME`).

Timestamp structure. Databases can structure timestamps in different ways: The value for a date–time timepoint (such as `12/20/96 3:30 PM`) can be stored in a single attribute, or the components may be stored as a separate date attribute (with value `12/20/96`) along with a time attribute (with value `3:30 PM`).

These three types of representational mismatches correspond to three standard classes of mismatches that Kamel and Zviran [1] call scale conflict, name conflict, and structural conflict, respectively. Prior methods to resolve such data heterogeneity are applicable to these types of temporal mismatches in clinical databases. We shall not discuss these well-defined solutions in this paper, but they are part of our computer-based method.

Semantic Mismatch

Semantic mismatch occurs if the timestamps for two sets of data are related in semantics, but are not equal.

Timestamp method. Most databases store a single timestamp for events and a pair of timestamps for states. Some databases may not time stamp certain data elements. Thus, not all timestamping approaches fit a canonical interval-based representation.

Temporal granularity. The timestamps for different clinical datum can be represented at different time units (such as day, hour, and minute). We cannot directly compare, however, timestamps of different time units, because the former do not form a complete linear ordering. Many database applications require such an ordering as a prerequisite for temporal querying.

Temporal coding. Temporal information for clinical data may not be encoded to reflect the data's period of temporal validity. For example, one database may model the start and end timestamps for a patient's episode of chest pain, whereas another database may store instead the start time of the chest pain along with its duration. The latter timestamping scheme does not create a data representation that permits direct interval-based comparisons.

Temporal dimension. For some schemas, a timestamp measures the datum's time of occurrence in the real world, whereas, for other schemas, a timestamp represents the time the value for the datum was entered into the system. The former timestamp captures *valid time*, whereas the latter stores *transaction time*. With the exception of data from real-time electronic data collection, these two timestamps do not have identical values. For temporal reasoning in decision support, we must distinguish which temporal dimension we need.

Unlike representational mismatches, these four types of semantic differences have not been addressed adequately by past approaches. This paper focuses on the resolution of semantic mismatches, and shows how a formal model of timestamps can encode and map across such differences.

A FOUNDATIONAL MODEL OF TIME

We present in this section a temporal model that can represent the complexity of timestamping semantics. We then use this model to encode formally the various legacy timestamping methods.

Timeline Model

The uniform timestamping scheme that we have proposed for Chronus is based on a discrete model of time. For our current research, we start instead with a temporal model that permits time to be a continuous linear variable; such values of time are equal to the real numbers. Figure 1 shows the resulting timeline.

We can divide such a timeline into equally sized time periods the duration of which varies by how fine or coarse we wish to make a discrete time unit [6]. We make the finest discretization equal to the smallest time-unit level (such as a second) at which we can represent timepoints in a database; as Figure 1 indicates, the set of finest time units on the timeline is isomorphic to the set of integers on the real line. To represent the next coarser time-unit level (such as a minute), we group a fixed number of the finest time units into a timepoint at that time-unit level. For example, the first minute (minute 0) on a timeline is a grouping of the first 60 time units at the level of seconds. We continue to define coarser time-unit level (such as an hour) as a grouping of time units at the finer time-unit level. This temporal representation thus establishes a hierarchical relationship between values of different time-unit levels.

With the timeline model, we can define primitive temporal elements. A *timepoint* is the pair (t, μ), where t is an integer value at a time unit μ. An *interval* is a set of adjacent timepoints; we represent this temporal element as the triplet (t_1, t_2, μ), where t_1 and t_2 are the values that bound the time period at time unit μ. An example timepoint is (2, hour), which is the third hour from the start of our timeline (the first hour is at 0). According to the hierarchical time-unit representation, this timepoint would encompass the interval (120, 179, minute), or the 121st to the 180th minutes on the timeline.

We define a *duration* to be the distance on the timeline between two timepoints of the same time-unit level. A duration is thus a pair, (d, μ), in which d is a distance along the timeline measured at time unit μ. For example, the duration between the timepoints (7, day) and (4, day) is (3, day). We allow d to be either positive or negative depending on which direction we measure the distance. A *range* corresponds to a triplet (d_1, d_2, μ) such that d_1 and d_2 are the distance values on a timeline that bound a span of temporal durations at time unit μ. With this notation, we specify an example range of 2 to 4 hours as (2, 4, hour).

For any particular data element, the values of these temporal elements can be based on timelines along either the valid- or transaction-time dimensions. Also, we can transform these integer-based values into corresponding real-world calendar–date values through labeling functions that we do not detail here.

Timestamp Representations

With these formal definitions of time elements, we can specify precisely the different timestamping schemes of clinical databases. We categorize these approaches based on whether clinical data are events or states.

Event Timestamping. The timestamping of events (instantaneous data) occurs by one of four approaches in the legacy databases that we examined.

1. Timepoint. In this common approach, a database schema stamps an event with a single timepoint attribute that captures the instant at which the value occurred. We represent such a timestamp with the timepoint (t, μ). For some timestamping schemes, the value of the time unit is stored explicitly; in such cases, the value of μ can vary for each value of t. More commonly, μ will have one value for all timestamps of that data element. For example, a database with a timestamp called `VISIT_DATE` has day as an implicit time unit.

2. Timepoint–duration pair. This approach stamps an event's occurrence as a duration offset from a reference timepoint; an age attribute is a prime example of this approach. We can model such an attribute as a person's birthdate timestamp by combining a duration (d, μ), where d is the age value and μ is set at year, with a timepoint (t, μ), which corresponds to the transaction timestamp at which the age value was stored in the database. Thus, the duration (37, year) stored with the timepoint $(t,$ day) represents that a person is 37 years of age at that reference timepoint.

3. Interval. This approach permits two timepoint attributes as bounds for the time period during which an event took place. A user thus has an *interval of uncertainty* to capture an event's occurrence; consequently, she could store the timing of a heart attack within an 8-hour interval, instead of at a coarser timepoint value (such as day). We represent this timestamping method by an interval (t_1, t_2, μ), where t_1 and t_2 are the values of the start and stop timepoints, respectively, and μ is their time-unit level.

4. Nontemporal. As we noted previously, some database may store no timestamps for an event. To allow such events to be queried, we can provide a zero-length interval, which has endpoints equal to a single value (such as the current timepoint), or we can create an interval of uncertainty (such as from negative infinity to now) to provide temporal indeterminacy.

State Timestamping. For states (data that have a significant duration), we find three timestamping schemes in legacy databases.

1. Interval. In this representation, the temporal dimensions of the state are stored as a pair of timepoint attributes. Each of these timepoints is actually an event. For example, the first timepoint of a medication state captures the event of administering the drug,

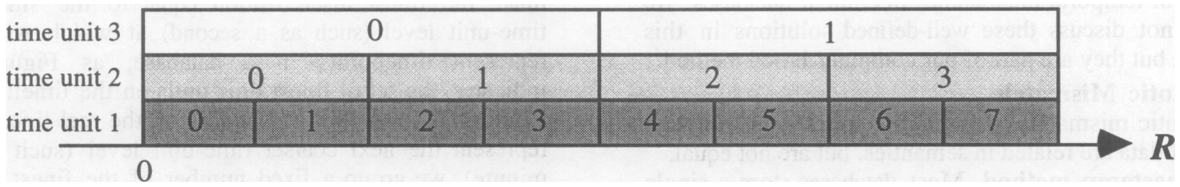

Figure 1. A timeline that shows the relationship between time as a continuous value and as sets of discretized timepoints at various time-unit levels. Time is modeled here on the real line; as shown, these time values are divisible into sets of equally distant units. This figure shows a hypothetical set of time units, in which the finest time unit (time unit 1) is the set of smallest divisions of the timeline (enumerated as timepoints numbered 0 through 7). We can group the timepoints at time unit 1 into timepoints at a coarser time-unit levels. Thus, timepoint 0 at time unit 2 encompasses timepoints 0 and 1 at time-unit level 1. These hypothetical time-unit divisions do not correspond to standard calendar–date granularities, but the example figure demonstrates the hierarchical grouping of any discrete set of timepoint values into time units.

whereas the second timepoint represents the event of discontinuing the drug. Thus, we can specify this timestamping approach as a pair of timepoints, [(t_1, μ_1), (t_2, μ_2)]. The time units of these two timepoints do not need to be equal.

2. Timepoint–duration pair. This type of state timestamping stores the temporal validity of a state as a pair of timepoint and duration attributes. For example, a database may model the time period of a patient's sore throat as occurring for 36 hours prior to a clinic visit on day t on a timeline. We represent this timestamp by the timepoint–duration pair [(t, day), (-36, hour)]. As in this example, the duration value can be negative, and the time units of the pair do not have to be equivalent.

3. Nontemporal. In this approach, no timestamps are stored explicitly for a state in the database. As with events that have no timestamps, we require a representation that makes explicit the temporal validity of such data. We use a pair of timepoints, and assign the temporal values appropriate timepoint constants (such as negative infinity and now).

We use these formal notations of the timestamping methods of events and states to encode precisely the temporal dimensions of data in legacy databases. Once we have established how a particular databases stores timestamps for each data element, we can use mapping functions to transform these data with temporal mismatches into the canonical temporal representation.

TEMPORAL MAPPING FUNCTIONS

As noted, our Chronus system allows applications to formulate temporal queries over a uniform temporal representation that is strictly interval based. Thus, each data element specified in a temporal query must be stamped with an interval (t_1, t_2, μ), where the value of the time unit μ is set at the query level [7]. For event- and state-based data in the database that do not have this particular timestamp representation, we define mapping functions to overcome temporal mismatches prior to performing the temporal query. We outline in this section the seven mapping functions that are needed to resolve semantic mismatches among temporal data. These functions are based on our timeline model and time-unit hierarchy.

1. Timepoint truncate. To transform a timepoint at a time unit that is finer than the query's time unit, we ascend the time-unit hierarchy to find the value of the coarser timepoint. We define the function timepoint-truncate((t, μ_1), μ_2), where μ_1 is finer than μ_2, to return the timepoint value at μ_2 that includes the timepoint (t, μ_1). For example, timepoint-truncate ((137, minute), hour) returns (2, hour), as this minute value occurs during the third hour.

2. Timepoint expand. The hierarchical representation of the time units in Figure 1 shows that a timepoint value at one time-unit level consists of an interval of timepoints at a finer level. We use the function timepoint-expand((t, μ_1), μ_2), where μ_1 is coarser than μ_2, to return the interval (u_1, u_2, μ_2) that denotes the range of timepoints that comprise t at the μ_2 time-unit level. With this function, we can transform timepoints at time units coarser than the canonical time unit. Thus, timepoint-expand((0, hour), minute) equals the interval (0, 59, minute).

3. Interval approximate. After mapping with the timepoint-expand function a timepoint at one time-unit level into an interval at a finer level, we may wish to have only a single timepoint at the finer time-unit level. By this approximation, applications can create a uniform temporal view that does not have the intervals of uncertainty created by the timepoint-expand function. We define the function interval-approximate((u_1, u_2, μ), p), where (u_1, u_2, μ) is an interval and p is a value between 0 and 100, to return the timepoint(t, μ), such that t is equal to or closest to the value that is p percentage of the interval. For example, interval-approximate((120, 179, minute), 50) returns the timepoint (150, minute), because this value is at the 50th percentage of the duration, or at the midpoint, of the interval (120, 179, minute).

4. Interval truncate. This function uses the semantics of the timepoint-truncate function to map the endpoints of intervals between various time-unit levels.

5. Interval expand. This function applies the timepoint-expand semantics to interval endpoints.

6. Duration convert. In addition to the time-unit transformations of timepoints and of intervals, we can also map between durations of different time-unit levels. If a duration value d_1 is measured precisely, we can convert this value into a duration value d_2 at a finer time unit. The function duration-convert((d, μ_1), μ_2), where μ_1 is coarser than μ_2, returns the product of d and the number of timepoints at μ_2 that comprise a timepoint at μ_1. For example, duration-convert((3, minute), second) equals the duration (180, second).

7. Duration expand. Some duration values in a database are not precise distance measurements. For example, an age value in a database that equals 37 years does not imply an exact age for the patient; this value is true both for a patient who became 37 on the date that information was gathered and for a patient who is one day less than 38 years in age. For such *imprecise* durations, we transform their values into a range at a finer time-unit level. The function duration-expand((d, μ_1), μ_2), where μ_2 is finer than μ_1, returns the range (d_1, d_2, μ_2) such that d_1 and d_2 are the bounds for d at the coarser time-unit level. Thus, duration-expand((37, year), month) equals the range (444, 455, month).

For data elements in a database that have semantic mismatch in their temporal dimensions, we may need more than one of these seven functions to transform such data into the uniform temporal representation. For example, to convert a timepoint–duration

representation of the attribute age into an interval-based format, we may need two functions. We use the duration-expand function to convert the age in years into a range of days, and we subtract each of the range endpoints from the reference date at which the database stored the age. The result is a year-long interval of days that bounds the birthdate. If we need instead a single-date approximation of the birthdate as the start of a person's life, we use next the interval-approximate function to return a timepoint of time-unit day.

DISCUSSION

In this paper, we have presented a foundational model of time, and have shown that it can unify different timestamping schemes present in heterogeneous clinical database. We have implemented this model by extending our Chronus querying method [4]. We call the extended system Synchronus, and it has two new components: (1) a *temporal metaschema* that allows developers to encode the timestamping method for underlying data in the database; and (2) *mapping operators* the semantics of which are based on our temporal mapping functions. For each data element that an application specifies in a temporal query, Synchronus checks the metaschema to find its encoded timestamping scheme. The system then applies the mapping operators to overcome any mismatches in the temporal dimensions of the base data. The resulting data are uniformly interval stamped, so we can then use the Chronus querying method on such data to verify the temporal conditions in the query.

A straightforward comparison of our research with other formal, published work on the modeling of temporal data in clinical databases reveals that Synchronus is novel in its ability to query existing timestamping schemes. Our approach thus supports the goals of integrating clinical data in heterogeneous databases and of creating decision-support applications that have a uniform interface to existing databases. Other researchers [8, 9] have relied instead on new database technologies to create highly expressive temporal representations, so that users can model explicitly the varying semantics of timestamps at the database level. To use the complex temporal model in the object-oriented methods of Combi et al. [8] or in the constraint propagation network of Dolin [9] requires the transfer and the reconfiguration of data into these database schemas.

A closer comparison of our research with other approaches reveals that we provide applications a simpler temporal model for querying. A fundamental assumption of our work is that we can satisfy the temporal querying needs of applications with a canonical interval-based schema that has timestamps at a single time-unit level. In contrast, the temporal querying method by Combi et al. [8] requires that applications handle varying time units at the query level, whereas applications that use the temporal representation of Dolin must manage the intervals of uncertainty present in its temporal scheme. Although we have shown that we can extend our querying method to permit such features [4], we do not find a wide variety of applications that need such capabilities for querying. Thus, our work focuses on establishing sufficient expressivity rather than temporal complexity in querying clinical data. We are currently testing Synchronus within decision-support systems, and we plan to verify in future papers that our foundational temporal model is adequate for a variety of clinical applications.

Acknowledgments

This work has been supported in part by grant HS06330 from the Agency for Health Policy and Research, and by grants LM05208, LM05708, and LM07033 from the National Library of Medicine.

REFERENCES

1. Kamel, M.N., and Zviran, M. Heterogeneous database integration in a hospital information systems environment. *Fifteenth Annual SCAMC*. Washington, DC. P.D. Clayton (ed), McGraw-Hill. 1991, pp. 363–7.
2. Sujansky, W., and Altman, R. Towards a standard query model for sharing decision-support applications. *Eighteenth Annual SCAMC*. Washington, DC. J.G. Ozbolt (ed), Hanley & Belfus. 1994, pp. 325–31.
3. Wiederhold, G., Jajodia, S., and Litwin, W. Integrating temporal data in a heterogeneous environment. In *Temporal Databases: Theory, Design, and Implementation*, A.U. Tansel Clifford, J., Gadia, S., et al., (eds). 1993. Redwood City, CA: Benjamin/Cummings.
4. Das, A.K., and Musen, M.A. A temporal query system for protocol-directed decision support. *Methods of Information in Medicine*, 1994. 33: 358–370.
5. Kahn, M.G. Modeling time in medical decision support programs. *Medical Decision Making*, 1991. 11: 249–64.
6. Wang, X.S., Jajodia, S., and Subrahmanian, V.S. Temporal modules: An approach toward federated temporal databases. *1993 ACM SIGMOD*. Washington, DC. ACM. 1993, pp. 227–36.
7. Das, A.K. and Musen, M.A. A comparison of the temporal expressiveness of three database query methods. *Nineteenth Annual SCAMC*. New Orleans, LA. R.M. Gardner (ed), AMIA. 1995, pp. 325–31.
8. Combi, C., Pinciroli, F., Cavallaro, M., and Cucchi, G. Querying temporal clinical databases with different time granularities. *Nineteenth Annual SCAMC*. New Orleans, LA. R.M. Gardner (ed), AMIA. 1995, pp. 326–30
9. Dolin, R.H. Modeling the temporal complexities of symptoms. *Journal of the American Medical Informatics Association*, 1995. 2: 323–31.